**NAME: AKSHAT HITEN SHAH**

**ENROLLMENT NUMBER: 200410107043**

**SUBJECT: DATA STRUCTURES AND ALGORITHMS**



**#include <stdio.h>**

**void swap(int , int);**

**int main()**

**{**

**printf("Akshat shah\t 200410107043\n");**

**int a = 50;**

**int b = 100;**

**printf("Before swapping the values in main a = %d, b = %d\n",a,b);**

**swap(a,b);**

**printf("After swapping values in main a = %d, b = %d\n",a,b);**

**}**

**void swap (int a, int b)**

**{**

**int temp;**

**temp = a;**

**a=b;**

**b=temp;**

**printf("After swapping values in function a = %d, b = %d\n",a,b);**

**}**

**OUTPUT:-**

**A screenshot of a computer

Description automatically generated with medium confidence**



**#include<stdio.h>**

**void change(int \*num) {**

**printf("Before adding value inside function num=%d \n",\*num);**

**(\*num) += 100;**

**printf("After adding value inside function num=%d \n", \*num);**

**}**

**int main() {**

**int x=100;**

**printf("Akshat Shah\t 200410107043\n");**

**printf("Before function call x=%d \n", x);**

**change(&x);**

**printf("After function call x=%d \n", x);**

**return 0;**

**}**

**OUTPUT:-**

**Text

Description automatically generated**



**Program:**

**#include <stdio.h>**

**#include <stdlib.h>**

**struct stack{**

**int size;**

**int top;**

**int\*S;**

**};**

**//crating stack**

**void create(struct stack\*st){**

**printf("enter the size ");**

**scanf("%d",&st->size);**

**st->top=-1;**

**st->S=(int\*)malloc(st->size\*sizeof(int));**

**}**

**//push function**

**void push(struct stack\*st,int x){**

**if(st->top==st->size-1)**

**printf("stack overflow\n");**

**else{**

**st->top++;**

**st->S[st->top]=x;**

**}**

**}**

**//displaying stack**

**void display(struct stack st){**

**for(int i=st.top;i>=0;i--){**

**printf("%d ",st.S[i]);**

**}**

**// printf("\n");**

**}**

**//pop function**

**int pop(struct stack\*st){**

**int x=-1;**

**if(st->top==-1)printf("stack underflow\n");**

**else{**

**x=st->S[st->top--];**

**}**

**return x;**

**}**

**//peek function**

**int peep(struct stack st,int pos){**

**int x=-1;**

**if(st.top-pos+1<0)**

**printf("invalid position");**

**else{**

**x=st.S[st.top-pos+1];**

**}**

**return x;**

**}**

**//to check whether stack is empty or not**

**int isEmpty(struct stack st){**

**if(st.top==-1)**

**return 1;**

**return 0;**

**}**

**//to check whether stack is full or not**

**int isFull(struct stack st){**

**return st.top=st.size-1;**

**}**

**//value of top of the stack**

**int stackTop(struct stack st){**

**if(!isEmpty(st))**

**return st.S[st.top];**

**return -1;**

**}**

**int main()**

**{ printf("Akshat shah\t 200410107043\n");**

**struct stack st;**

**create(&st);**

**push(&st,10);**

**push(&st,20);**

**push(&st,30);**

**push(&st,40);**

**push(&st,50);**

**//pop(&st);**

**printf("peeped value is %d\n",peep(st,2));**

**printf("top of the stack is %d\n",stackTop(st));**

**display(st);**

**return 0;**

**}**

**Output: -**

**Text

Description automatically generated**



**#include <stdio.h>**

**#include <stdlib.h>**

**#include<string.h>**

**struct node{**

**char data;**

**struct node\*next;**

**}\*top=NULL;**

**void push(char x){**

**struct node\*t;**

**t=(struct node\*)malloc(sizeof(struct node));**

**if(t==NULL)**

**printf("stack is full");**

**else{**

**t->data=x;**

**t->next=top;**

**top=t;**

**}**

**}**

**char pop(){**

**struct node\*t;**

**char x=-1;**

**if(top==NULL)**

**printf("stack is empty");**

**else{**

**t=top;**

**top=top->next;**

**x=t->data;**

**free(t);**

**}**

**return x;**

**}**

**void display(){**

**struct node\*p;**

**p=top;**

**while(p){**

**printf("%d ",p->data);**

**p=p->next;**

**}**

**printf("\n");**

**}**

**int isBalanced(char \*exp){**

**for(int i=0;exp[i]!='\0';i++){**

**if(exp[i]=='(')**

**push(exp[i]);**

**else if(exp[i]==')'){**

**if(top==NULL)**

**return 0;**

**pop();**

**}**

**}**

**if(top==NULL)**

**return 1;**

**else**

**return 0;**

**}**

**int pre(char x){**

**if(x=='+' || x=='-'){**

**return 1;}**

**else if(x=='\*' || x=='/')**

**return 2;**

**return 0;**

**}**

**int isOperand(char x){**

**if(x=='+'|| x=='-'|| x=='\*'|| x=='/')**

**return 0;**

**else**

**return 1;**

**}**

**char\*inToPost(char \*infix){**

**int i=0,j=0;**

**char\*postfix;**

**int len=strlen(infix);**

**postfix=(char\*)malloc((len+2)\*sizeof(char));**

**while(infix[i]!='\0'){**

**if(isOperand(infix[i]))**

**postfix[j++]=infix[i++];**

**else{**

**if(pre(infix[i])>pre(top->data))**

**push(infix[i++]);**

**else{**

**postfix[j++]=pop();**

**}**

**}**

**}**

**while(top!=NULL)**

**postfix[j++]=pop();**

**postfix[j]='\0';**

**return postfix;**

**}**

**int main()**

**{**

**char\*infix="a+b\*c";**

**push('#');**

**char\*postfix=inToPost(infix);**

**printf("Akshat Shah\t 200410107043\n");**

**printf("postfix expression is %s ",postfix);**

**// char\*exp="((a+b)+(c-d))";**

**// printf("%d ",isBalanced(exp));**

**// push(10);**

**// push(20);**

**// push(30);**

**// push(40);**

**// push(50);**

**// display();**

**return 0;**

**}**

**OUTPUT:-**

**Text

Description automatically generated**



**#include <stdio.h>**

**#include <stdlib.h>**

**#define n 100**

**int f = -1, r = -1, queue[n];**

**int main ()**

**{printf("Akshat Shah\t 200410107043\n");**

**while(1)**

**{**

**printf("Enter choice:\n1)insert\n2)delete\n3)exit\n");**

**int i,choice;**

**printf("\nEnter your choice:\n");**

**scanf("%d",&choice);**

**switch(choice)**

**{**

**case 1:**

**insert();**

**printf("elements in queue are\t");**

**for(i=f;i<=r;i++){**

**printf("%d ",queue[i]);**

**}**

**break;**

**case 2:**

**delete();**

**printf("elements in queue are\t");**

**for(i=f;i<=r;i++){**

**printf("%d ",queue[i]);**

**}**

**break;**

**case 3:**

**exit(0);**

**break;**

**}**

**}**

**return 0;**

**}**

**void insert()**

**{**

**int item;**

**if (r == n - 1)**

**{**

**printf ("\noverflow");**

**}**

**if (f == -1)**

**{**

**f = r = 0;**

**}**

**else{**

**r = r + 1;**

**}**

**printf ("\nInserted element:");**

**scanf ("%d", &item);**

**queue[r]=item;**

**}**

**void delete()**

**{**

**int ditem;**

**if (f == -1)**

**{**

**printf ("underflow");**

**}**

**else**

**ditem = queue[f];**

**if (f == r)**

**{**

**f = r = -1;**

**}**

**else**

**{**

**f = f + 1;**

**}**

**printf("\nitem deleted:");**

**}**

**OUTPUT:-**

**Text

Description automatically generated**



**#include <stdio.h>**

**#include <stdlib.h>**

**#define n 10**

**int f = -1, r = -1, queue[n];**

**int main ()**

**{printf("Akshat Shah\t 200410107043\n");**

**while(1)**

**{**

**printf("Enter choice:\n1)insert\n2)delete\n3)display\n4)exit\n");**

**int i,choice;**

**printf("\nEnter your choice: ");**

**scanf("%d",&choice);**

**switch(choice)**

**{**

**case 1:**

**insert();**

**break;**

**case 2:**

**delete();**

**break;**

**case 3:**

**display();**

**break;**

**case 4:**

**exit(0);**

**}**

**}**

**return 0;**

**}**

**void insert()**

**{**

**int item;**

**if(f==-1 && r==-1)**

**{**

**f=0;**

**r=0;**

**queue[r]=item;**

**}**

**else if((r+1)%n==f)**

**{**

**printf("Queue is overflow..");**

**}**

**else**

**{**

**r=(r+1)%n;**

**queue[r]=item;**

**}**

**printf("\ninserted item:");**

**scanf("%d",&item);**

**queue[r]=item;**

**}**

**void display(){**

**int i;**

**if(f<=r)**

**for(i=f;i<=r;i++)**

**printf("%d ",queue[i]);**

**else{**

**for(i=f;i<=n;i++)**

**printf("%d ",queue[i]);**

**for(i=1;i<=r;i++)**

**printf("%d ",queue[i]);**

**}**

**}**

**void delete(){**

**int ditem;**

**if(f==-1){**

**printf("\nstack underflow");**

**return;**

**}**

**ditem=queue[f];**

**if(f==r)**

**f=r=-1;**

**else if(f==n-1)**

**f=0;**

**else**

**f=f+1;**

**}**

**OUTPUT:-**

**Text

Description automatically generated**



**#include <stdio.h>**

**#include <stdlib.h>**

**struct node**

**{**

**int data;**

**struct node \*next;**

**} \*first;**

**//creating linked list**

**// void create (int A[], int n)**

**// {**

**// int i;**

**// struct node \*t, \*last;**

**// first = (struct node \*) malloc (sizeof (struct node));**

**// first->data = A[0];**

**// first->next = NULL;**

**// last = first;**

**// for (i = 1; i <= n; i++)**

**// {**

**// t = (struct node \*) malloc (sizeof (struct node));**

**// t->data = A[i];**

**// t->next = NULL;**

**// last->next = t;**

**// last = t;**

**// }**

**// }**

**//display function**

**void display (struct node \*p)**

**{**

**while (p)**

**{ //here while(p) means while(p!=NULL)**

**printf ("%d ", p->data);**

**p = p->next;**

**}**

**}**

**//count the number of nodes**

**int count (struct node \*p)**

**{**

**int c = 0;**

**while (p)**

**{**

**c++;**

**p = p->next;**

**}**

**return c;**

**}**

**//sum of the data in nodes**

**int sum (struct node \*p)**

**{**

**int add = 0;**

**while (p)**

**{**

**add += p->data;**

**p = p->next;**

**}**

**return add;**

**}**

**//average of the data in nodes**

**float avg (struct node \*p)**

**{**

**float a = 0;**

**while (p)**

**{**

**a = sum (first) / count (first);**

**p = p->next;**

**}**

**return a;**

**}**

**//Reverse a linked list-sliding pointers**

**void Reverse(struct node \*p)**

**{**

**struct node \*q=NULL,\*r=NULL;**

**while(p!=NULL)**

**{**

**r=q;**

**q=p;**

**p=p->next;**

**q->next=r;**

**}**

**first=q;**

**}**

**//deleting node**

**int delete(struct node\*p,int index){**

**struct node\*q;**

**int x=-1,i;**

**if(index<1||index>count(first))**

**return x;**

**if(index==1){**

**q=first;**

**x=q->data;**

**first=first->next;**

**free(q);**

**return x;**

**}**

**else{**

**for(i=0;i<index-1;i++){**

**q=p;**

**p=p->next;**

**}**

**q->next=p->next;**

**x=p->data;**

**free(p);**

**return x;**

**}**

**}**

**//inserting in a linked list**

**void insert(struct node\*p,int index,int x){**

**struct node\*t;**

**int i;**

**if(index<0||index>count(first))**

**return;**

**t=(struct node\*)malloc(sizeof(struct node));**

**t->data=x;**

**if(index==0){**

**t->next=first;**

**first=t;**

**}**

**else{**

**for(i=0;i<index-1;i++)**

**p=p->next; //here only p=p->next is in the loop**

**t->next=p->next;**

**p->next=t;**

**}**

**}**

**int main ()**

**{ printf("Akshat Shah\t 200410107043\n");**

**// int A[] = { 10, 20, 30, 40, 50, 60 };**

**// create (A,5);**

**//insert(first,1,80);**

**insert(first,0,24);**

**insert(first,1,80);**

**insert(first,2,67);**

**insert(first,3,98);**

**// insert(first,2,78);**

**// insert(first,3,68);**

**display (first);**

**// printf ("\ncount is %d\n", count (first));**

**// printf ("sum is %d\n", sum (first));**

**// printf ("avg is %f\n", avg (first));**

**//Reverse(first);**

**// display(first);**

**// printf("\ndeleted item is %d\n",delete(first,2));**

**//display(first);**

**return 0;**

**}**

**OUTPUT:-**

**Text

Description automatically generated**



#include <stdio.h>

#include <stdlib.h>

struct node{

int data;

struct node\*next;

}\*top=NULL;

void push(int x){

struct node\*t;

t=(struct node\*)malloc(sizeof(struct node));

if(t==NULL)

printf("stack is full");

else{

t->data=x;

t->next=top;

top=t;

}

}

int pop(){

struct node\*t;

int x=-1;

if(top==NULL)

printf("stack is empty");

else{

t=top;

top=top->next;

x=t->data;

free(t);

}

return x;

}

void display(){

struct node\*p;

p=top;

while(p){

printf("%d ",p->data);

p=p->next;

}

printf("\n");

}

int main()

{printf("Akshat Shah\t 200410107043\n");

push(10);

push(20);

push(30);

push(40);

push(50);

pop();

display();

return 0;

}

OUTPUT:-

Text

Description automatically generated



#include <stdio.h>

#include <stdlib.h>

struct Node

{

int data;

struct Node \*next;

}\*front=NULL,\*rear=NULL;

void enqueue(int x)

{

struct Node \*t;

t=(struct Node\*)malloc(sizeof(struct Node));

if(t==NULL)

printf("Queue is FUll\n");

else

{

t->data=x;

t->next=NULL;

if(front==NULL)

front=rear=t;

else

{

rear->next=t;

rear=t;

}

}

}

int dequeue()

{

int x=-1;

struct Node\* t;

if(front==NULL)

printf("Queue is Empty\n");

else

{

x=front->data;

t=front;

front=front->next;

free(t);

}

return x;

}

void Display()

{

struct Node \*p=front;

while(p)

{

printf("%d ",p->data);

p=p->next;

}

printf("\n");

}

int main()

{printf("Akshat Shah\t 200410107043\n");

enqueue(10);

enqueue(20);

enqueue(30);

enqueue(40);

enqueue(50);

printf("Queue is:");

Display();

dequeue();

printf("After dequeue\n");

//printf("%d ",dequeue());

Display();

return 0;

}

OUTPUT:-

Text

Description automatically generated



#include <stdio.h>

#include <stdlib.h>

struct node{

struct node\*prev;

int data;

struct node\*next;

}\*first=NULL;

//creating doubly linked list

void create(int A[],int n){

struct node\*last,\*t;

first=(struct node\*)malloc(sizeof(struct node));

first->data=A[0];

first->prev=NULL;

first->next=NULL;

last=first;

for(int i=1;i<n;i++){

t=(struct node\*)malloc(sizeof(struct node));

t->data=A[i];

t->prev=last;

t->next=last->next;

last->next=t;

last=t;

}

}

//displaying doubly linked list

int display(struct node\*p){

while(p){

printf("%d ",p->data);

p=p->next;

}

printf("\n");

}

//counting number of nodes in doubly linked list

int count(struct node\*p){

int c=0;

while(p){

c++;

p=p->next;

}

return c;

}

//inserting elements in a circular linked list

void insert(struct node\*p,int pos,int x){

struct node\*t;

if(pos<0||pos>count(first))

return;

if(pos==0){

t=(struct node\*)malloc(sizeof(struct node));

t->data=x;

t->prev=first->prev;

first->prev=t;

t->next=first;

first=t;

}

else{

for(int i=0;i<pos-1;i++)p=p->next;

t=(struct node\*)malloc(sizeof(struct node));

t->data=x;

t->prev=p;

t->next=p->next;

if(p->next)p->next->prev=t;

p->next=t;

}

}

//deleting node from doubly linked list

int delete(struct node\*p,int pos){

int x=-1;

if(pos<0||pos>count(first))

return -1;

if(pos==1){

p=first;

first=first->next;

x=p->data;

if(first)first->prev=p->prev;

free(p);

return x;

}

else{

for( int i=0;i<pos-1;i++)p=p->next;

p->prev->next=p->next;

if(p->next)

p->next->prev=p->prev;

x=p->data;

free(p);

return x;

}

}

//reversing a doubly linked list

void reverse(struct node\*p){

struct node\*q=NULL,\*r=NULL;

p=first;

q=p;

r=q;

p=p->next;

}

int main()

{printf("Akshat Shah\t 200410107043\n");

int A[]={10,20,30,40,50};

create(A,5);

printf("number of nodes are %d\n",count(first));

display(first);

printf("deleted node is %d\n",delete(first,5));

//insert(first,5,57);

display(first);

return 0;

}

OUTPUT:-

Text

Description automatically generated



#include <stdio.h>

#include <stdlib.h>

struct node{

int data;

struct node\*next;

}\*head;

//creating circular linked list

void create(int A[],int n){

int i;

struct node\*t,\*last;

head=(struct node\*)malloc(sizeof(struct node));

head->data=A[0];

head->next=head;

last=head;

for(i=1;i<n;i++){

t=(struct node\*)malloc(sizeof(struct node));

t->data=A[i];

t->next=last->next;

last->next=t;

last=t;

}

}

//displaying circular linked list

void display(struct node \*h)

{

do

{

printf("%d ",h->data);

h=h->next;

}while(h!=head);

printf("\n");

}

//display using recursion

void rdisplay(struct node\*h){

static int flag=0;

if(h!=head||flag==0){

flag=1;

printf("%d ",h->data);

rdisplay(h->next);

}

flag=0;

}

//counting number of nodes in circular linked list

int count(struct node\*p){

int c=0;

do{

c++;

p=p->next;

}while(p!=head);

return c;

}

//inserting in circular linked list

void insert(struct node\*p,int index,int x){

struct node\*t;

int i;

if(index<0||index>count(p))

return ;

if (index==0){

t=(struct node\*)malloc(sizeof(struct node));

t->data=x;

if(head==NULL){

head=t;

head->next=head;

}

else{

while(p->next!=NULL)p=p->next;

p->next=t;

t->next=head;

head=t;

}

}

else

{

for(i=0;i<index-1;i++)p=p->next;

t=(struct node\*)malloc(sizeof(struct node));

t->data=x;

t->next=p->next;

p->next=t;

}

}

//deleting from circular linked list

int delete(struct node\*p,int pos){

struct node\*q;

int x;

if(pos<1||pos>count(head))

return 0;

if(pos==1){

while(p->next!=head)p=p->next;

x=p->data;

if(head==NULL){

free(head);

head=NULL;

}

else{

p->next=head->next;

free(head);

head=p->next;

}

}

else{

for(int i=0;i<pos-2;i++)p=p->next;

q=p->next;

p->next=q->next;

x=q->data;

free(q);

}

return x;

}

int main()

{

printf("Akshat Shah\t 200410107043\n");

int A[]={10,20,30,40,50,60};

create(A,6);

rdisplay(head);

//printf("\nnumber of nodes are %d\n",count(head));

//insert(head,0,80);

printf("\ndeleted item is %d\n",delete(head,2));

display(head);

return 0;

}

OUTPUT:-

Text

Description automatically generated



#include <stdio.h>

void swap (int \*x, int \*y)

{

int temp = \*x;

\*x = \*y;

\*y = temp;

}

void bubble (int A[], int n)

{

int i, j;

for (i = 0; i < n - 1; i++)

{

int flag = 0;

for (j = 0; j < n - i - 1; j++)

{

if (A[j] > A[j + 1])

{

swap (&A[j + 1], &A[j]);

flag = 1;

}

}

if (flag == 0)

break;

}

}

int main ()

{printf ("Akshat Shah\t 200410107043\n");

int A[] = { 4, 7, 9, 13, 56, 35, 76, 8, 2, 1 }, n = 10;

bubble(A,10);

for (int i = 0; i < n; i++)

{

printf ("%d ", A[i]);

}

return 0;

}

OUTPUT:-

Text

Description automatically generated

**Practical-13**

**Aim:** Quick sort **Program:**

#include <stdio.h>

#include<stdlib.h>

void swap(int \*x,int \*y)

{

int temp=\*x; \*x=\*y;

\*y=temp;

}

int partition(int A[],int l,int h)

{

int pivot=A[l]; int i=l,j=h;

do

{

do{i++;}while(A[i]<=pivot); do{j--;}while(A[j]>pivot);

if(i<j)swap(&A[i],&A[j]);

}while(i<j);

swap(&A[l],&A[j]); return j;

}

void QuickSort(int A[],int l,int h)

{ int j;

if(l<h)

{

j=partition(A,l,h); QuickSort(A,l,j);

QuickSort(A,j+1,h);

}

}

int main()

{printf("Akshat Shah\t 200410107043\n"); int A[]={11,13,7,12,16,9,24,5,10,3},n=10,i;

QuickSort(A,0,10);

for(i=0;i<10;i++) printf("%d ",A[i]); printf("\n");

return 0;

}

OUTPUT:

Text

Description automatically generated



#include<stdio.h>

#include<stdlib.h>

void merge (int A[], int l, int mid, int h)

{

int i = l, j = mid + 1, k = l; int B[100]; while (i <= mid && j <= h)

{ if (A[i] < A[j]) B[k++] = A[i++];

else

B[k++] = A[j++];

}

for (; i <= mid; i++) B[k++] = A[i]; for (; j <= h; j++) B[k++] = A[j]; for (i = l; i <= h; i++) A[i] = B[i];

}

void IMergeSort (int A[], int n)

{ int p, l, h, mid, i; for (p = 2; p <= n; p = p \* 2)

{

for (i = 0; i + p - 1 < n; i = i + p)

{ l = i; h = i + p - 1; mid = (l + h) / 2; merge (A, l, mid, h);

} if (n - i > p / 2)

{ l = i; h = i + p - 1; mid = (l + h) / 2; merge (A, l, mid, n - 1);

}

}

if (p / 2 < n)

{

merge (A, 0, p / 2 - 1, n - 1);

}

}

int main ()

{

printf("Akshat Shah\t 200410107043\n");

int A[] = { 11, 5, 14, 2, 6, 3, 1 }, n = 7, i;

IMergeSort (A, n);

for (i = 0; i < n; i++)

printf ("%d ", A[i]);

printf ("\n"); return 0;

}

OUTPUT:

Text

Description automatically generated



#include <stdio.h>

int main()

{

printf("Akshat Shah\t 200410107043\n");

int i, low, high, mid, n, key, array[100];

printf("Enter number of elements:");

scanf("%d",&n);

printf("Enter %d integers:", n); for(i = 0; i < n; i++)

scanf("%d",&array[i]);

printf("Enter value to find:");

scanf("%d", &key); low = 0; high = n - 1; mid = (low+high)/2;

while (low <= high) { if(array[mid] < key)

low = mid + 1;

else if (array[mid] == key)

{

printf("%d found at location %d", key, mid+1); break;

}

else high = mid - 1; mid = (low + high)/2;

}

if(low > high)

printf("Not found! %d isn't present in the list.n", key); return 0;

}

OUTPUT:

Text

Description automatically generated